

Wykonał: Jakub Nowakowski

### 1. Cele Laboratorium

Laboratorium miało na celu zapoznanie studentów z podstawowym algorytmem triangulacji wielokątów monotonicznych oraz sprawdzaniem monotoniczności wielokątów.

### 2. Konfiguracja stanowiska

Procesor: intel Core i7-8750H

Wersja Python'a: Python 3.7.4

### 3. Wstęp teoretyczny

Podczas tego laboratorium do sprawdzania monotoniczności wielokątów została wykorzystana właściwość, że wielokąty Y-monotoniczne nie posiadają wierzchołków dzielących i łączących. Do triangulacji wielokąta został wykorzystany podstawowy algorytm przedstawiony na wykładzie.

### 4. Opis działania programu

#### a) Procedura sprawdzania monotoniczności wielokąta

Program iteruje po wierzchołkach wielokąta nadając im oznaczenia, zgodnie z następującymi warunkami:

- Wierzchołek jest końcowy, gdy obaj sąsiedzi są powyżej a kąt wewnętrzny  $< \pi$
- Wierzchołek jest dzielący, gdy obaj sąsiedzi są poniżej a kąt wewnętrzny  $> \pi$
- Wierzchołek jest łączący, gdy obaj sąsiedzi są powyżej a kąt wewnętrzny  $< \pi$
- Wierzchołek jest początkowy, gdy obaj sąsiedzi są poniżej a kąt wewnętrzny  $< \pi$
- Wierzchołek jest prawidłowy, gdy ma jednego sąsiada powyżej drugiego poniżej

Jeżeli podczas iterowania napotka wierzchołek dzielący lub łączący wtedy klasyfikuje wielokąt jako nie Y-monotoniczny.

#### b) Rozdział wielokąta na łańcuch lewy i prawy

Przy założeniu, że wierzchołki zostały wprowadzone w kierunku przeciwnym do kierunku wskazówek zegara, możemy rozdzielić je na podstawie prostej właściwości:

-wierzchołki leżące pomiędzy wierzchołkiem najniższym, a najwyższym należą do łańcucha prawego

-wierzchołki leżące pomiędzy wierzchołkiem najwyższym, a najniższym należą do lewego łańcucha

Rozdzielenie na dwa łańcuchy odbywa się w programie wykorzystując slicing, przy sprawdzeniu czy wierzchołek najniższy znajduje się w tablicy przed najwyższym czy po nim.

### c) Triangulacja wielokąta

Po określeniu, które wierzchołki należą do którego łańcucha, sortujemy je malejąco według współrzędnej y-kowej. Następnie inicjalizujemy stos i umieszczamy na nim dwa najwyżej położone wierzchołki. Teraz iterując się po kolejnych wierzchołkach dodajemy krawędzie tworzące trójkąty. Warunki dodawania krawędzi są następujące:

-Jeśli aktualnie rozpatrywany wierzchołek znajduje się na innym łańcuchu niż szczyt stosu łączymy go z wszystkimi wierzchołkami na stosie, a na stosie zostawiamy dwa ostatnio analizowane wierzchołki  
-Jeśli wierzchołek znajduje się na tym samym łańcuchu co szczyt stosu, mamy do rozpatrzenia dwa przypadki:

- utworzony trójkąt należałby do wielokąta wtedy usuwamy wierzchołek ze stosu

-nie należałby do wielokąta umieszczamy wtedy badane wierzchołki na stosie

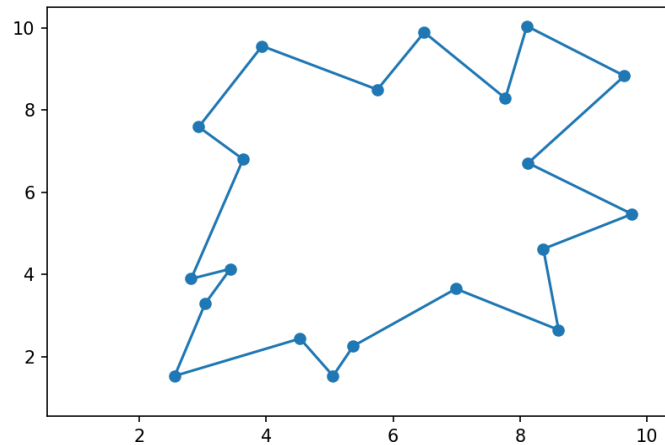
## 5. Wykorzystane struktury danych

W programie wielokąt jest przechowywany jako lista wierzchołków. Taka implementacja wielokąta umożliwia nam zachowanie względnej kolejności kolejnych wierzchołków oraz ułatwia operowanie na wierzchołkach (sortowanie i rozdział na łańcuchy). Po Rozdzieleniu wielokąta na łańcuchy każdy wierzchołek przechowuje dodatkowo informacje o tym w jakim łańcuchu się znajduje. Do przechowania siatki powstałych trójkątów wykorzystany jest zbiór który przechowuje krotki zawierające punkty. Taka implementacja eliminuje powtarzanie się tych samych elementów w siatce. Jednocześnie przechowywanie krotek zawierających punkty umożliwia nam pamiętanie względnej kolejności każdej składowej siatki.

## Wielokąty, na których były przeprowadzone testy

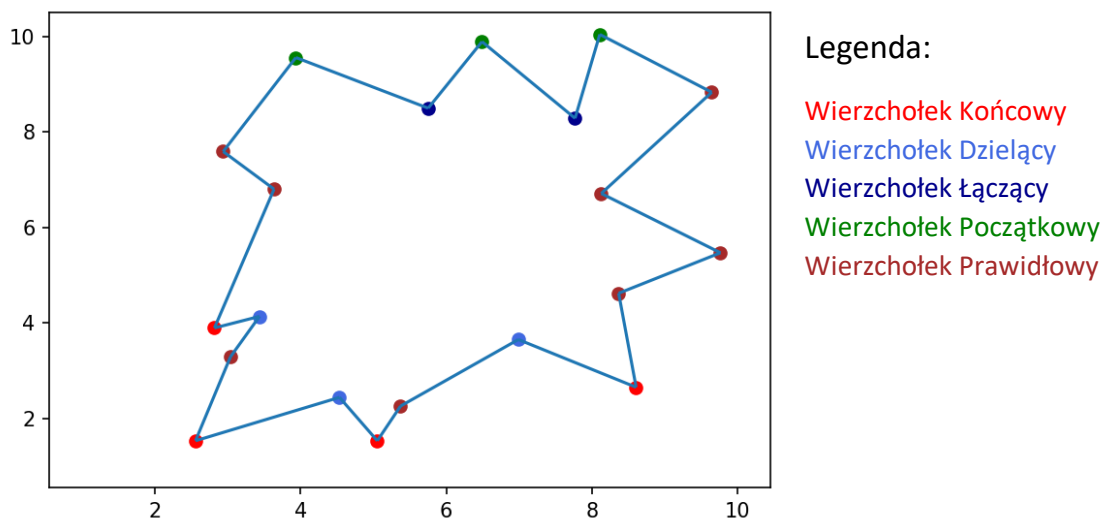
### a) Test kolorowania wierzchołków

Ta część algorytmu została przetestowana na wielokącie, który zawierał każdy typ wierzchołka:



Rysunek 1 Wielokąt 1

Kolorowanie wierzchołków podanego wielokąta według typów:

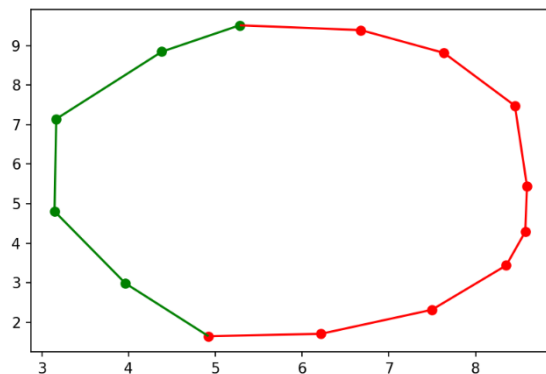


Rysunek 2 Kolorowanie wielokąta 1

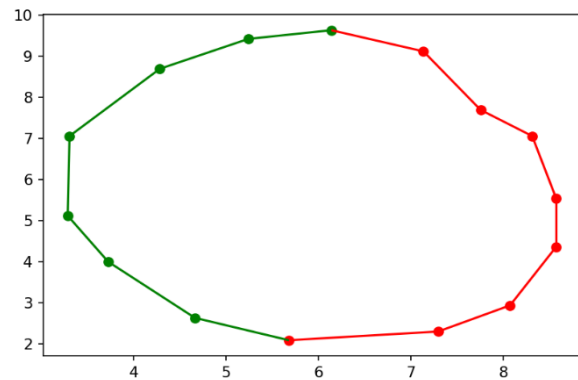
Wszystkie wierzchołki zostały pokolorowane prawidłowo niezależnie od typu, na tej podstawie można wnioskować, że implementacja kolorowania została przeprowadzona w sposób prawidłowy

### b) Rozdział na lewy i prawy łańcuch

Test rozdzielenia na lewy i prawy łańcuch został przeprowadzony dwa razy, dla wielokąta, w którym najpierw wprowadzono wierzchołek najwyższy, oraz dla wielokąta który wcześniej miał wprowadzony wierzchołek najniższy.



Rysunek 4 Podział na łańcuchy, gdy najpierw wprowadzony był najniższy wierzchołek



Rysunek 3 Podział na łańcuchy, gdy najpierw wprowadzony był najwyższy wierzchołek

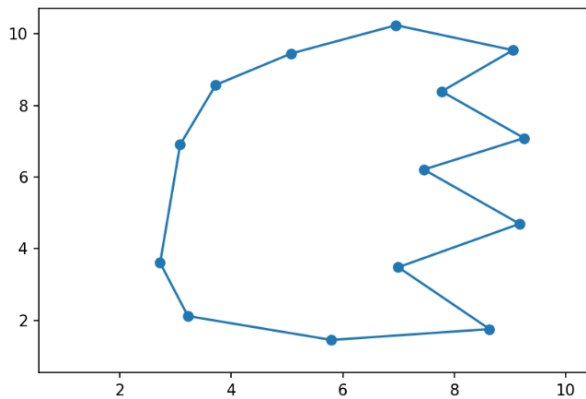
Lewy łańcuch został oznaczony na zielono prawy na czerwono. Jak widać niezależnie od kolejności wprowadzania wierzchołków, podział na łańcuch lewy i prawy odbywa się w prawidłowy sposób.

### c) Triangulacja

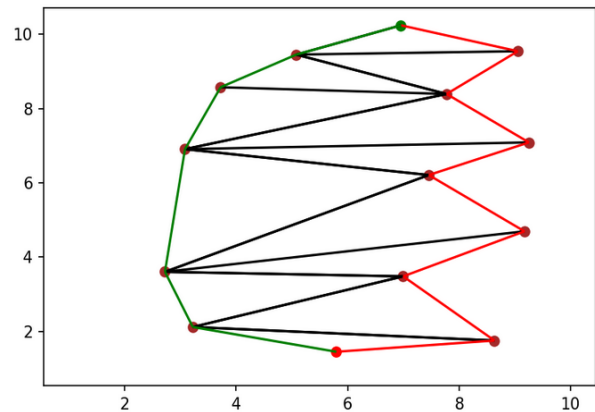
Triangulacja została przeprowadzona na różnych typach wielokątów, które powodowały różne sytuacje mogące spowodować niewłaściwy przebieg działania programu.

### Figura pierwsza

Wielokąt o tym kształcie został wybrany, aby sprawdzić sprawdzanie, poprawności działania algorytmu, gdy zdejmowane wierzchołki znajdują się na tym samym łańcuchu, a niektóre tworzone przez nie trójkąty i



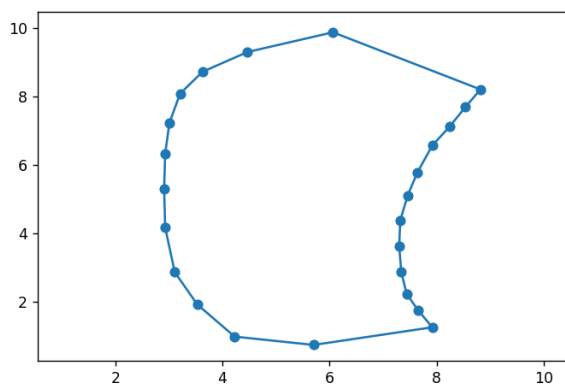
Rysunek 5 Omawiany wielokąt



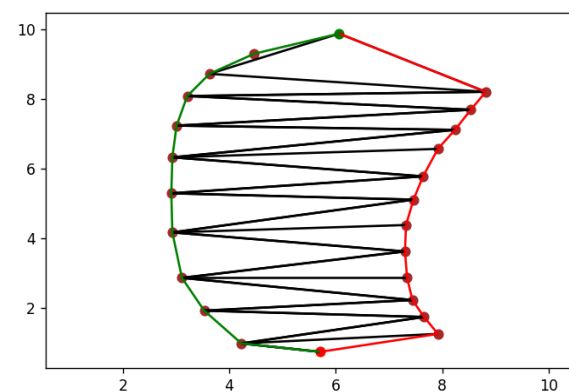
Rysunek 6 Triangulacja omawianego wielokąta

### Figura druga

Testuje sytuację analogiczną jak wyżej, gdy żaden utworzony trójkąt nie należy do wielokąta.



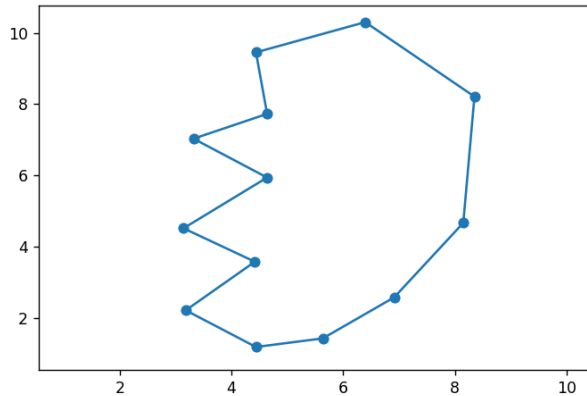
Rysunek 7 Omawiany wielokąt



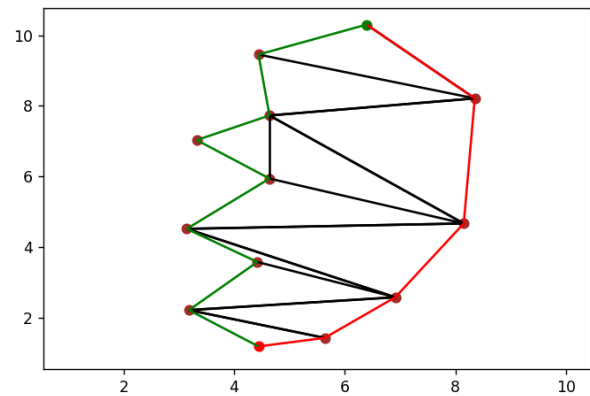
Rysunek 6 Triangulacja Omawianego wielokąta

### Figura Trzecia

Analogiczna sytuacja jak figura pierwsza, ale odbita względem osi oy.



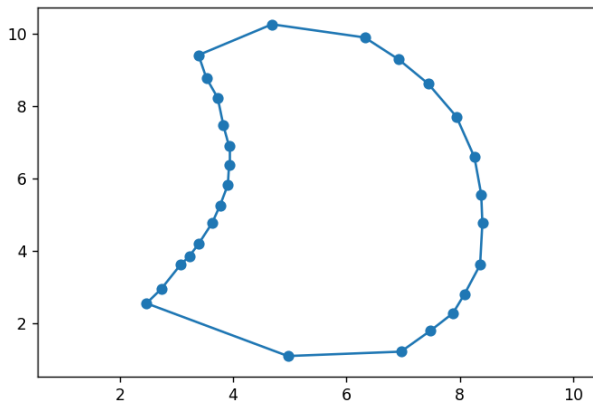
Rysunek 9 Omawiany wielokąt



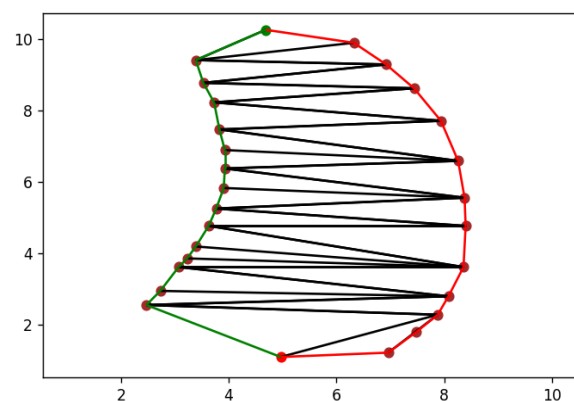
Rysunek 8 Triangulacja omawianego wielokąta

### Figura czwarta

Analogiczna sytuacja jak dla figury drugiej, również odbita względem osi oy.



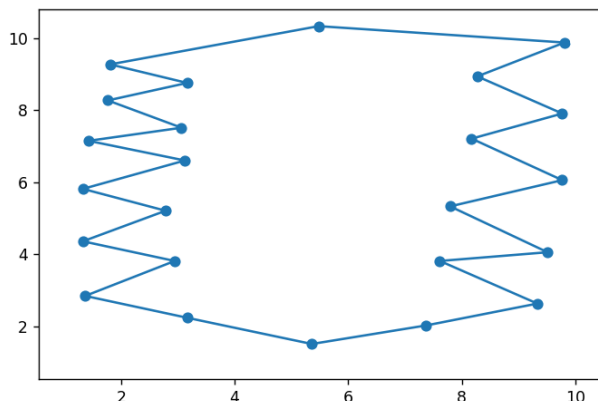
Rysunek 11 Omawiany wielokąt



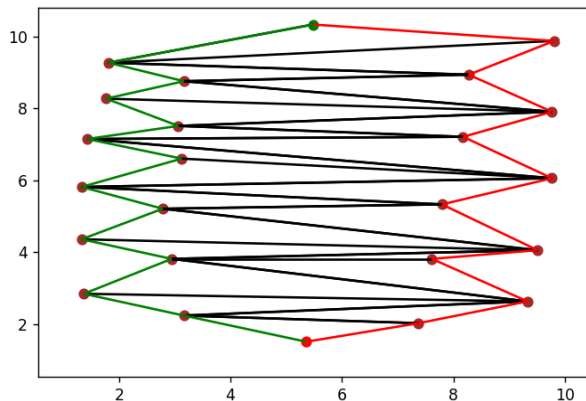
Rysunek 10 Triangulacja omawianego wielokąta

### Figura Piąta

Figura ta łączy problem z figury pierwszej i trzeciej.



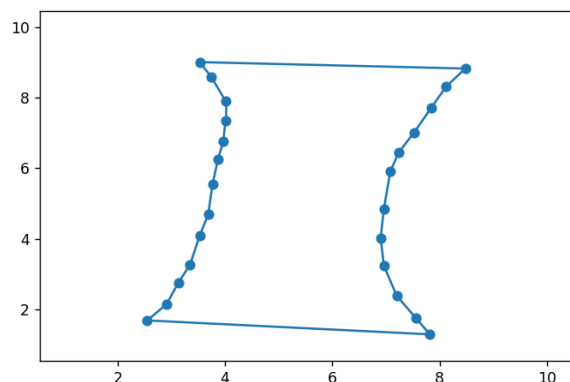
Rysunek 12 Omawiany Wielokąt



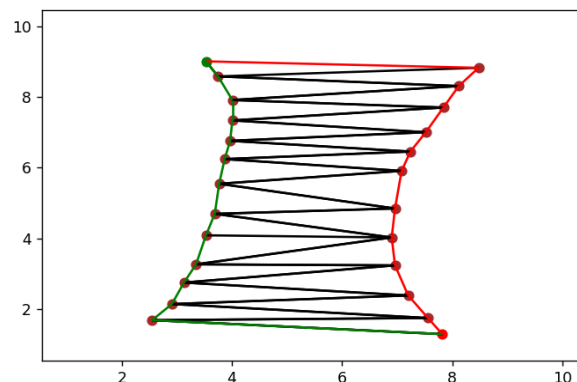
Rysunek 13 Triangulacja omawianego wielokąta

### Figura szósta

Połączenie figury czwartej i drugiej.



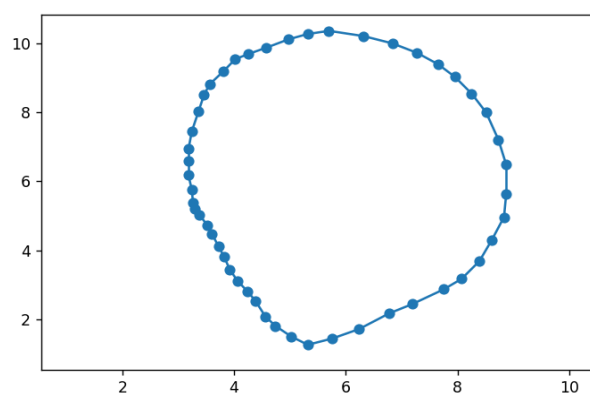
Rysunek 15 Omawiany wielokąt



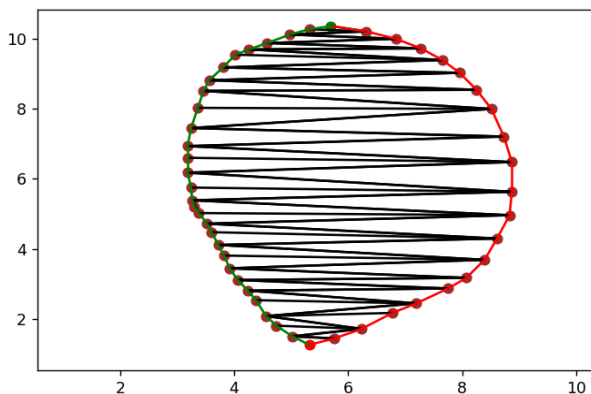
Rysunek 14 Triangulacja omawianego wielokąta

### Figura siódma

Ta figura ma na celu sprawdzenie prawidłowe przechodzenie obu łańcuchów w pętli głównej programu, gdy łańcuchy różnią się zagęszczeniem punktów.



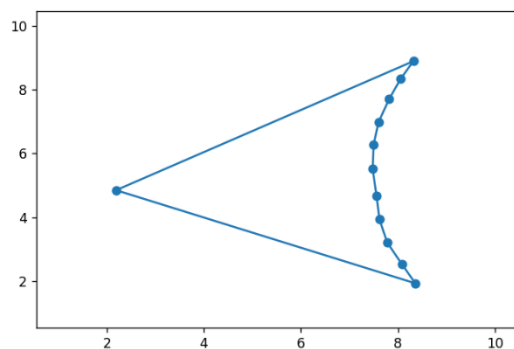
Rysunek 16 Omawiany wielokąt



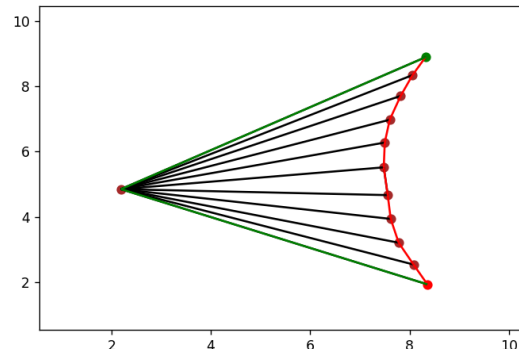
Rysunek 17 Triangulacja omawianego wielokąta

### Figura ósma

Trudność w tej figurze polega na tym że wszystkie trójkąty powstałe w triangulacji mają jeden wspólny wierzchołek.



Rysunek 17 Omawiany wielokąt



Rysunek 19 Triangulacja omawianego wielokąta

## 6. Podsumowanie

Po przetestowaniu programu na wyżej wymienionych zbiorach, nie zauważono błędów w działaniu programu można więc stwierdzić, że działa on w sposób poprawny.